Python 5 – Les fonctions dans Python

1 – Déclaration puis appel d'une fonction

Déclaration : Une fonction se déclare en début de programme à l'aide du mot-clé **def**. On place ensuite le nom de la fonction, les paramètres entre parenthèses et un double-point : .

Exemple :

```
def fois2(a):        → Déclaration de la fonction qui recevra un paramètre nommé localement a
    b = a * 2        → On crée une variable locale b qui sera supprimée à la fin de la fonction
    return b         → On renvoie le résultat et on sort
```

01° Placez ce programme dans Thonny, et lancez le. La fonction se place-t-elle en mémoire ? _____
La fonction se lance-t-elle seule ? _____

02° Faire ces **appels** à la fonction dans la console. La fonction se lance-t-elle cette fois ? _____

```
>>> fois2(10)
20
```
Lors de cet appel, que va contenir la variable-paramètre **a** ? _____

```
>>> fois2(40)
80
```
Lors de cet appel, que va contenir la variable-paramètre **a** ? _____

03° Tester cette version pour vérifier qu'elle fait bien la même chose (mais en plus simple encore).

```
def fois2(a):
    return a * 2
```

2 – Fonction à plusieurs paramètres

```
def mystere(a, b, c):      → Prototype de la fonction (3 paramètres locaux a, b et c)
    reponse = a  + b*2 + c → On crée une variable locale reponse qui sera supprimée à la fin
    return reponse         → On renvoie
```

04° Placez ce programme dans Thonny, et lancez le. La fonction se place-t-elle en mémoire ? _____
La fonction se lance-t-elle seule ? _____

05° Utilisez l'appel suivant dans la Console. Que vaut **a** lors de cet appel ? Que vaut **b** ? Que vaut **c** ?

```
>>> mystere(2, 10, 7)
29
```

06° Expliquer ce que doit renvoyer **mystere(10, 2, 3)** puis expliquer la dernière ligne de ceci :

```
>>> a = mystere(10, 2, 3)
>>> b = a * 10
>>> b
```

07° Compléter la fonction **addition():** elle doit renvoyer l'addition de deux contenus qu'on fournira lors de l'appel. Donner l'appel qu'on doit en faire pour additionner 10 et 15. Voici son prototype : **def addition(a, b):**

08° Compléter cette fonction pour renvoyer **a*x + b** à partir des paramètres a, x et b. **def affine(a, x, b):**

09° Fournir deux appels réalisés avec cette fonction, au choix.

10° Pour tester une **égalité** entre deux variables a et b, doit-on taper **a == b** ou **a != b** ?

11° Compléter la dernière ligne de **verification_v1()** pour renvoyer True si le paramètre **mot_recu** passé en paramètre est le même que le bon mot de passe, contenu dans une variable **mdp,** locale à la fonction.

```
def verification_v1(mot_recu):
    mdp = '1234!'
    return ???
```

12° Fournir deux appels réalisés avec cette fonction, au choix.

13° Utiliser la fonction ci-dessous pour vérifier qu'elle renvoie la même chose que la précédente. Un utilisateur peut-il se rendre compte que les deux codes sont différents ? _____

```
def verification_v2(mot_recu):
    mdp = '1234!'
    if mot_recu != mdp:
        return False
    else:
        return True
```

14° Modifier la fonction **verification_v2** pour que son code contienne une comparaison (==) et pas une différence (!=).

15° Créer une fonction **plusGrand(a,b)** qui renvoie True si a est plus grand que b, False sinon.